Coinspect

Source Code Review

Prepared for Swarm Fund • September 2017

V1.2

1. Table of Contents

2. Executive Summary

In September 2017, Swarm Fund Project engaged Coinspect to perform a security audit of the crowdsale, token and multi-sig wallet smart contracts, together with other inherited library contracts: OpenZeppelin and MiniMeToken. The contracts were retrieved from the Swarm github repository on August 31st, 2017. The objective of the audit was to evaluate the security of the crowdsale, ERC20-compatible token and multi-sig wallet implementations. During the assessment, Coinspect identified 1 high-risk issues, 0 medium-risk issues, and 6 low-risk issues.

All the important risks reported were mitigated in the audit commit: 89fde6d818a82133887add5f3acaf934fe1679d4.

3. Introduction

The contract **SwarmCrowdsale.sol** is a traditional crowdsale contract that changes the token price rate at providing steps, and being pausable, and therefore owned. It dynamically mints tokens in the **SwarmToken.sol** contract, which is a ERC20-compliant contract with vesting, minting and cloning functionality, using MiniMe.sol in its core. All tokens, either pre-allocated of bought during the token sale are vested. The vesting lasts 336 days, and it restricts the amount of funds that can be extracted linearly over this period, but in discrete steps of 42 days each. The token cloning functionality is provided by MiniMe. MiniMe is controlled by the **SwarmCrowdsale.sol** contract during the sale, and by a multi-signature wallet (**MultiSigWallet.sol**) after it is finalized. This wallet also receives all tokens that have not been sold during the crowdsale. Minime enables the controller to to pause and resume the trading of the token, claim other tokens or ether sent by mistake to the token contract, and generate more tokens at will.
All contracts use SafeMath methods to prevent overflows.

A whitebox security audit was conducted on this contract. The present report was completed on September 5th by Coinspect and includes results from the audit.

4. Findings

4.1. Wallet contract does not implement the TokenController interface

High Risk (RESOLVED)

When the crowdsale is over, the token controller is changed from the SwarmCrowdsale contract to the MultiSigWallet contract. The following code extracted from SwarmCrowdsale shows the change:

```
/**
 * Overrides Base Function.
 * Take any finalization actions here
 * Ends token minting on finalization
 */
 function finalization() internal whenNotPaused {

   // Handle unsold token logic
   transferUnallocatedTokens();

   // Complete minting and start vesting of token
```

```
      token.finishMinting();

      // Transfer ownership to the wallet
      token.changeController(wallet);
    }
```

But theMultiSigWallet not implement the TokenController interface (defined in TokenController.sol). This interface exposes three methods:

- function proxyPayment(address _owner) payable returns(bool);
  Called when `_owner` sends ether to the MiniMe Token contract

- function onTransfer(address _from, address _to, uint _amount) returns(bool);
  Notifies the controller about a token transfer allowing the controller to react if desired

- function onApprove(address _owner, address _spender, uint _amount) returns(bool);
  Notifies the controller about an approval allowing the controller to react if desired

Because MultiSigWallet does not implement them, the expected behaviour after the finalization of the crowdsale is that:

1. Any attempt to transfer the tokens after the token sale will fail because wallet does not implement the onTransfer() method.
2. Any attempt to approve the third party transfer of tokens will fail because wallet does not implement the onApprove() method.
3. All ether payments to the token contract will be rejected because the wallet does not implement proxyPayment().

Recommendations

- Because onTransfer() and onApprove() are unused, remove from the TokenController interface the methods onTransfer() and onApprove(), leaving only the method proxyPayment.
- Remove from the MiniMeToken.sol contract the calls to onTransfer and onApprove(),
- Make MultiSigWallet inherit the TokenController interface.

Fixed in the audit commit

onTransfer() and onApprove() were removed from the TokenController

4.2. Wallet contract owners can generate tokens after the crowdsale is over for future funding rounds

Low Risk

Generating tokens after the crowdsale is a feature of the Swarm platform described in the white paper. Because the MultiSigWallet becomes the controller of the SwarmToken after the crowdsale is over, the majority of wallet owners can approve a transaction that calls the GenerateTokens in MiniMeToken. This enables the creation of new tokens after the crowdsale is over. Also wallet owners can destroy tokens from any address using destroyTokens() or transfer tokens from one user to another with transferFrom() without the need for approval of the owner. While these two last operations are not specified in the white paper, the risks involved are not higher than enabling the new tokens generation operation. It must be noted that new tokens generated are not subject to the vesting period.

Recommendations

Warn token buyers and holders that the power to generate more tokens is an intended feature of the platform.

4.3. Use block timestamps instead of block numbers for crowdsale dates

Low Risk

Due to the Ethereum difficulty bomb, it is very hard to estimate block numbers. Given that token sales usually run over several days, the in-accuracy due to unknown block times is actually higher than the max 15 minutes practical gameable of the block.timestamp. Hence the use block.timestamp (or now) instead of block.number for the crowdsale contract is recommended. This change was adopted in the last version of OpenZeppelin crowdsale contract.

Recommendation

Do nothing. Given that the interval of the Swarm crowdsale is short (one month) there is no benefit from switching to timestamp based intervals.

4.4. Unclear field naming in FinalizableCrowdsale.sol

Low Risk

In FinalizableCrowdsale constructor, the block numbers for the start and end dates for the crowdsale are called: _startTime and _endTime, which is confusing.

Recommendation

If block numbers are to be user, they should better be called _startBlockNumber and _endBlockNumber to prevent confusion.

4.5. Create and reuse numerical constants in SwarmCrowdsale

Low Risk (RESOLVED)

The number of tokens to pre-mine in specified as 100 * 10**18. Constants should be specified separately, and reusing other constants to reduce chances of errors.

Recommendations

- Create a constant for the number of tokens pre-created
- USe TOKEN_DECIMALS instead of 10**18.

Fixed in audit commit

4.6. Unchecked subtraction in MiniMeVestedToken

Low Risk (RESOLVED)

The following line is not checked for overflow:

uint256 vestingPeriodsRemaining = VESTING_TOTAL_PERIODS - vestedPeriodsCompleted;

Even if it can be proven that no overflow can occur, it is recommended to use a checked subtraction to prevent underflow if the code is modified in a future revision and the invariant is broken.

Recommendations

Replace the line by:
uint256 vestingPeriodsRemaining =
VESTING_TOTAL_PERIODS.sub(vestedPeriodsCompleted);

Fixed in audit commit

4.7. Buy may fail if MAX_TOKEN_SALE_CAP is overpassed

Very low Risk

If a user tries to buy more tokens than the maximum allowed, an exception is raised, preventing the user to buy any tokens.

Recommendation

If the cap is exceeded for the amount requested , allow the user to buy the remaining amount.

5. Disclaimer

The present security audit is limited to smart contract code. It does not cover the technologies and designs related to these smart contracts, nor the frameworks and wallets that communicate with the contracts, nor the general operational security of the company behind this project. This document should not be read as investment advice or an offering of tokens.